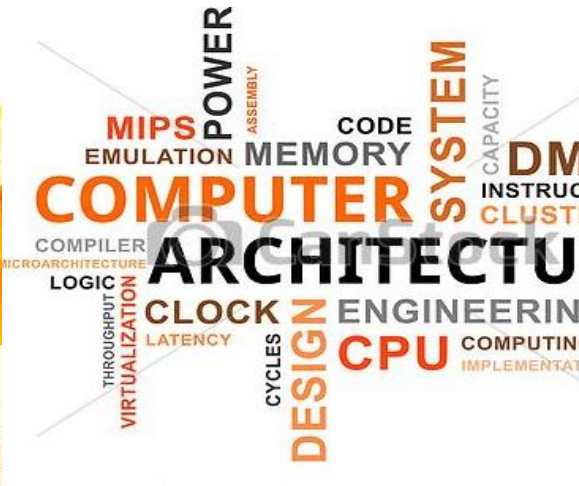
UNIT III
PROCESSOR AND PIPELINING

Fundamental concepts – Execution of a complete instruction – Multiple bus organization **– Hardwired control** – **Micro programmed control** – Pipelining: Basic concepts – Data hazards – Instruction hazards – Influence on Instruction sets – Data path and control consideration.

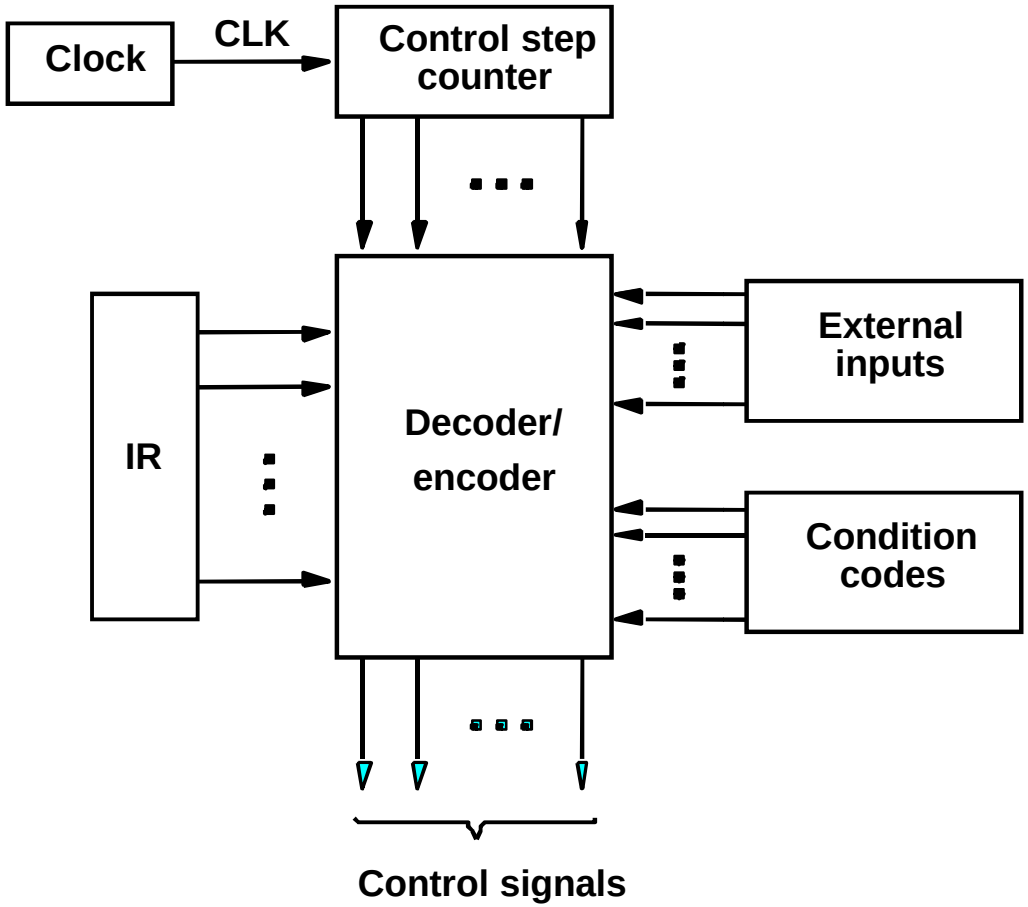


Recap the previous Class

Overview

- To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence.
- Two categories: hardwired control and microprogrammed control
- Hardwired system can operate at high speed; but with little flexibility.

Control Unit Organization



Detailed Block Description

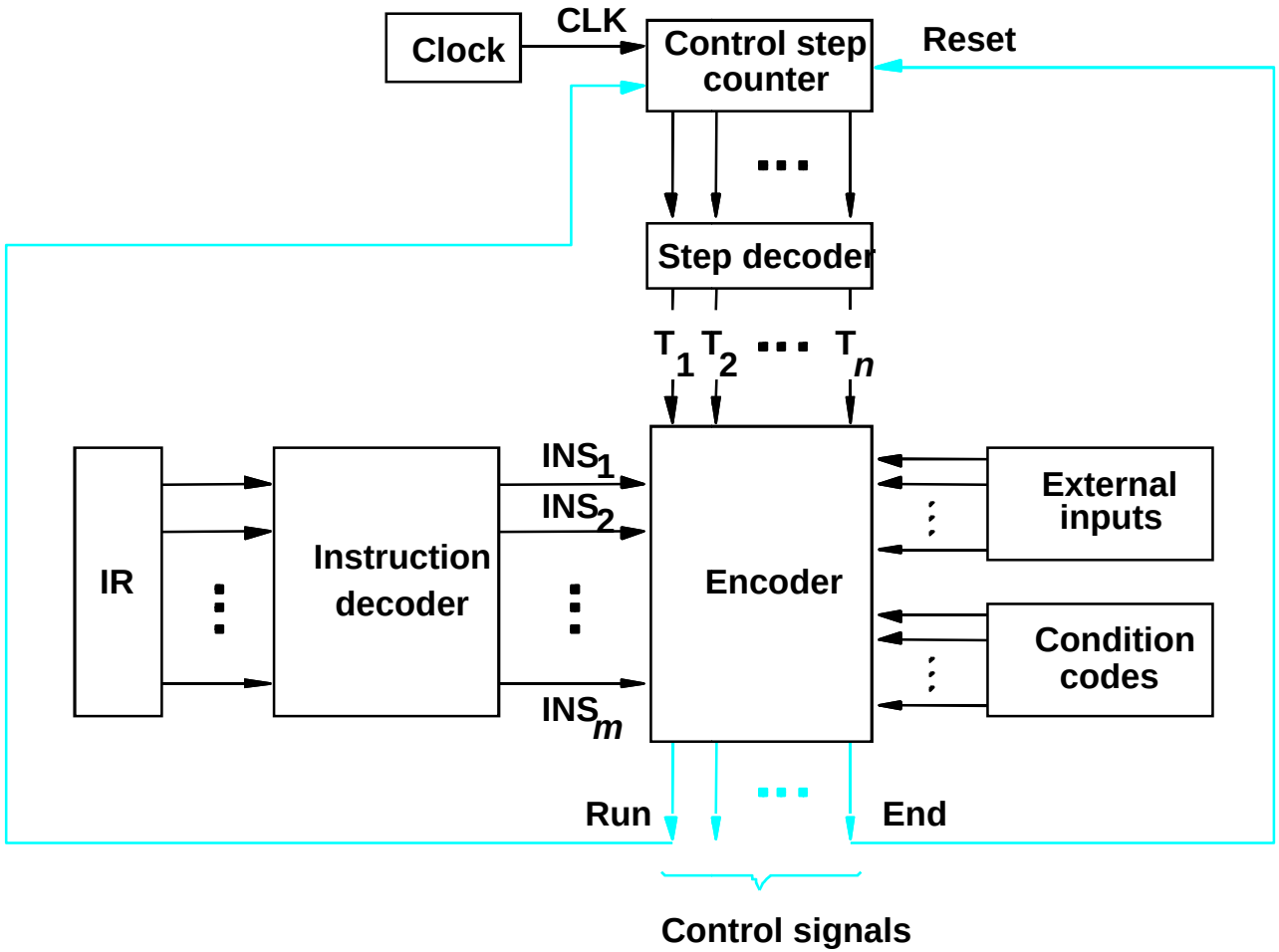


Figure : Separation of the decoding and encoding functions.

Generating Z_{in}

$$Z_{in} = T_1 + T_6 . ADD + T_4 . BR + ...$$

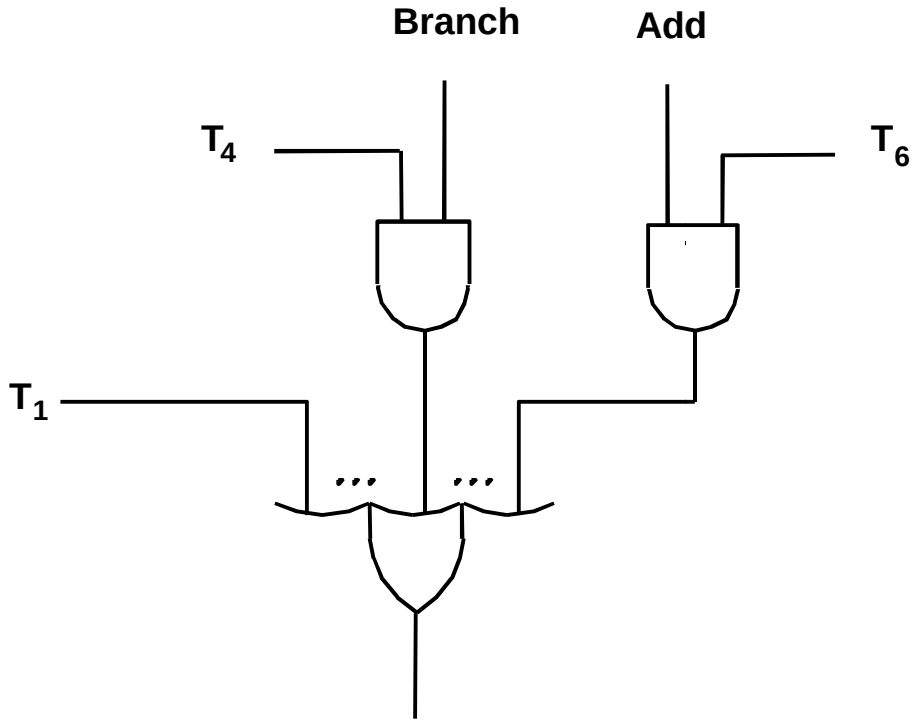


Figure :Generation of the Z_{in} control signal for the processor

Generating End

$$End = T_7 . ADD + T_5 . BR + (T_5 . N + T_4 . N) . BRN + ...$$

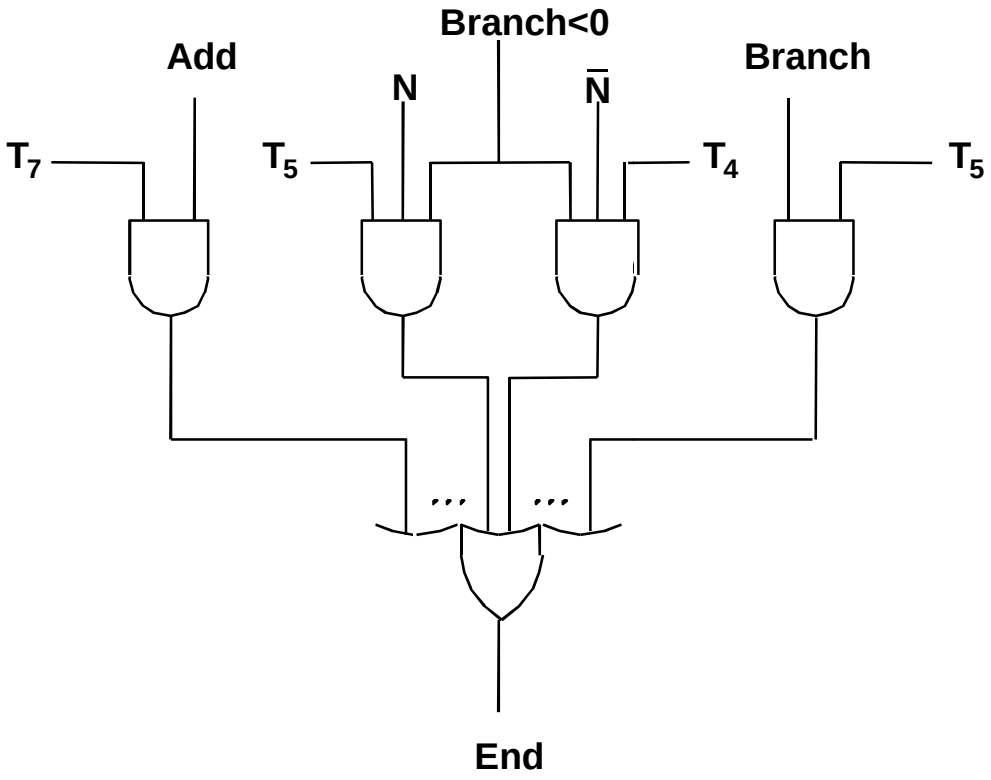


Figure: Generation of the End control signal.

A Complete Processor

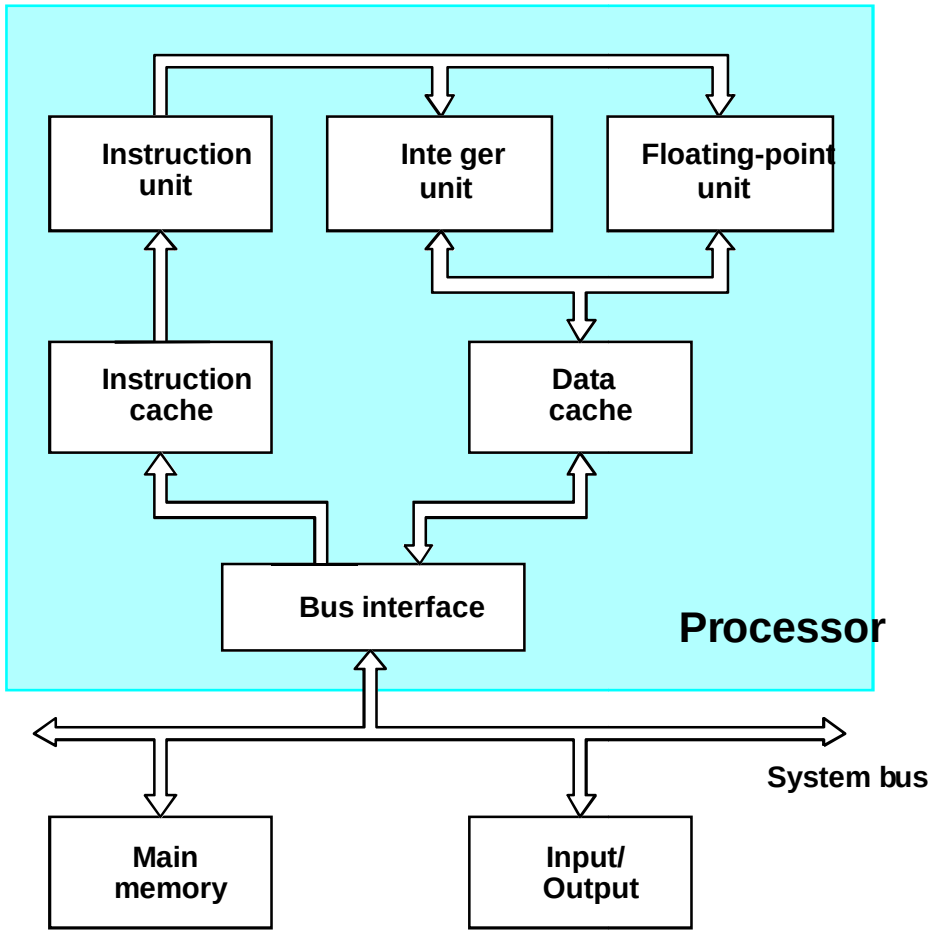


Figure: Block diagram of a complete processor

.

Microprogrammed Control

- Control signals are generated by a program similar to machine language programs.
- Control Word (CW); microroutine; microinstruction

Micro - instruction	..																..
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
4		0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	
5		0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	
6		0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	
7		0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	

Figure : An example of microinstructions.

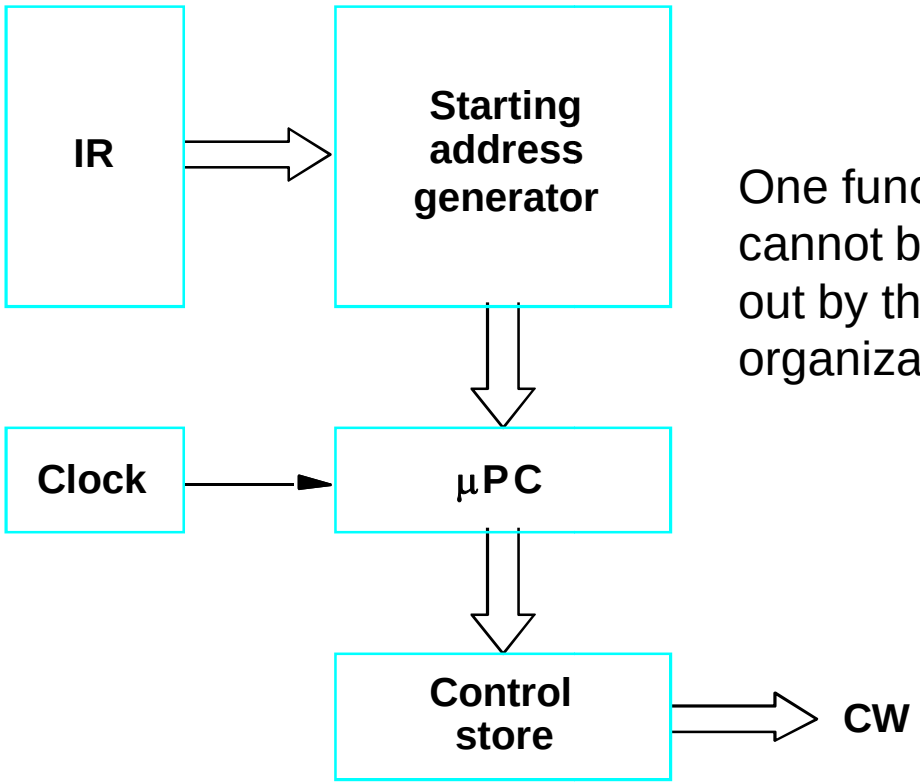
Overview

Step	Action
1	PC_{out} , MAR_{in} , Read, Select4, Add, Z_{in}
2	Z_{out} , PC_{in} , Y_{in} , WMF C
3	MDR_{out} , IR_{in}
4	$R3_{out}$, MAR_{in} , Read
5	$R1_{out}$, Y_{in} , WMF C
6	MDR_{out} , SelectY, Add, Z_{in}
7	Z_{out} , $R1_{in}$, End

Figure :Control sequence for execution of the instruction Add (R3),R1.

Basic organization of a microprogrammed control unit

- Control store



One function cannot be carried out by this simple organization.

Figure :. Basic organization of a microprogrammed control unit.

Conditional branch

- The previous organization cannot handle the situation when the control unit is required to check the status of the condition codes or external inputs to choose between alternative courses of action.
- Use conditional branch microinstruction.

Address	Microinstruction
0	PC_{out} , MAR_{in} , Read, Select4, Add, Z_{in}
1	Z_{out} , PC_{in} , Y_{in} , WMFC
2	MDR_{out} , IR_{in}
3	Branch to starting address of appropriate microroutine
. . . . .	
25	If N=0, then branch to microinstruction 0
26	Offset-field-of-IR_{out} , SelectY, Add, Z_{in}
27	Z_{out} , PC_{in} , End

Figure : Microroutine for the instruction Branch<0.

Microprogrammed Control

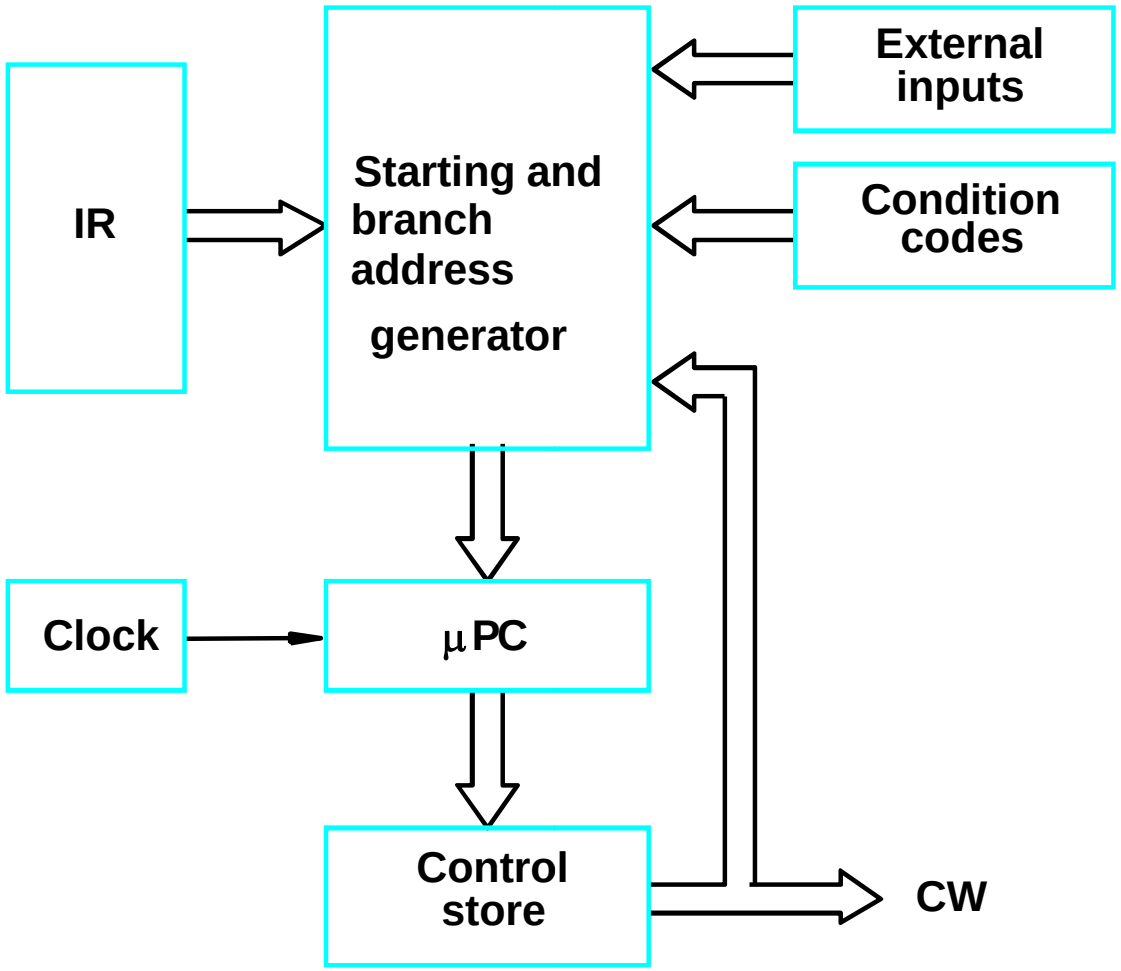


Figure :Organization of the control unit to allow conditional branching in the microprogram.

Microinstructions

- A straightforward way to structure microinstructions is to assign one bit position to each control signal.
- However, this is very inefficient.
- The length can be reduced: most signals are not needed simultaneously, and many signals are mutually exclusive.
- All mutually exclusive signals are placed in the same group in binary coding.

Partial Format for the Microinstructions

Microinstruction

F1	F2	F3	F4	F5

F1 (4 bits)	F2 (3 bits)	F3 (3 bits)	F4 (4 bits)	F5 (2 bits)
0000: No transfer	000: No transfer	000: No transfer	0000: Add	00: No action
0001: PC_{out}	001: PC_{in}	001: MAR_{in}	0001: Sub	01: Read
0010: MDR_{out}	010: IR_{in}	010: MDR_{in}	⋮	10: Write
0011: Z_{out}	011: Z_{in}	011: $TEMP_{in}$	1111: XOR	
0100: $R0_{out}$	100: $R0_{in}$	100: Y_{in}	16 ALU functions	
0101: $R1_{out}$	101: $R1_{in}$			
0110: $R2_{out}$	110: $R2_{in}$			
0111: $R3_{out}$	111: $R3_{in}$			
1010: $TEMP_{out}$				
1011: $Offset_{out}$				

F6	F7	F8	. . .

F6 (1 bit)	F7 (1 bit)	F8 (1 bit)
0: SelectY	0: No action	0: Continue
1: Select4	1: WMFC	1: End

What is the price paid for this scheme?

Require a little more hardware

Figure : An example of a partial format for field-encoded microinstructions.

Microprogram Sequencing

- If all microprograms require only straightforward sequential execution of microinstructions except for branches, letting a μPC governs the sequencing would be efficient.

Two disadvantages:

- Having a separate microroutine for each machine instruction results in a large total number of microinstructions and a large control store.
- Longer execution time because it takes more time to carry out the required branches.

Example: Add src, Rdst

- Four addressing modes: register, autoincrement, autodecrement, and indexed (with indirect forms).

Microinstructions with Next-Address Field

- A powerful alternative approach is to include an address field as a part of every microinstruction to indicate the location of the next microinstruction to be fetched.
- **Pros:** separate branch microinstructions are virtually eliminated; few limitations in assigning addresses to microinstructions.
- **Cons:** additional bits for the address field (around 1/6)

Microinstructions with Next-Address Field

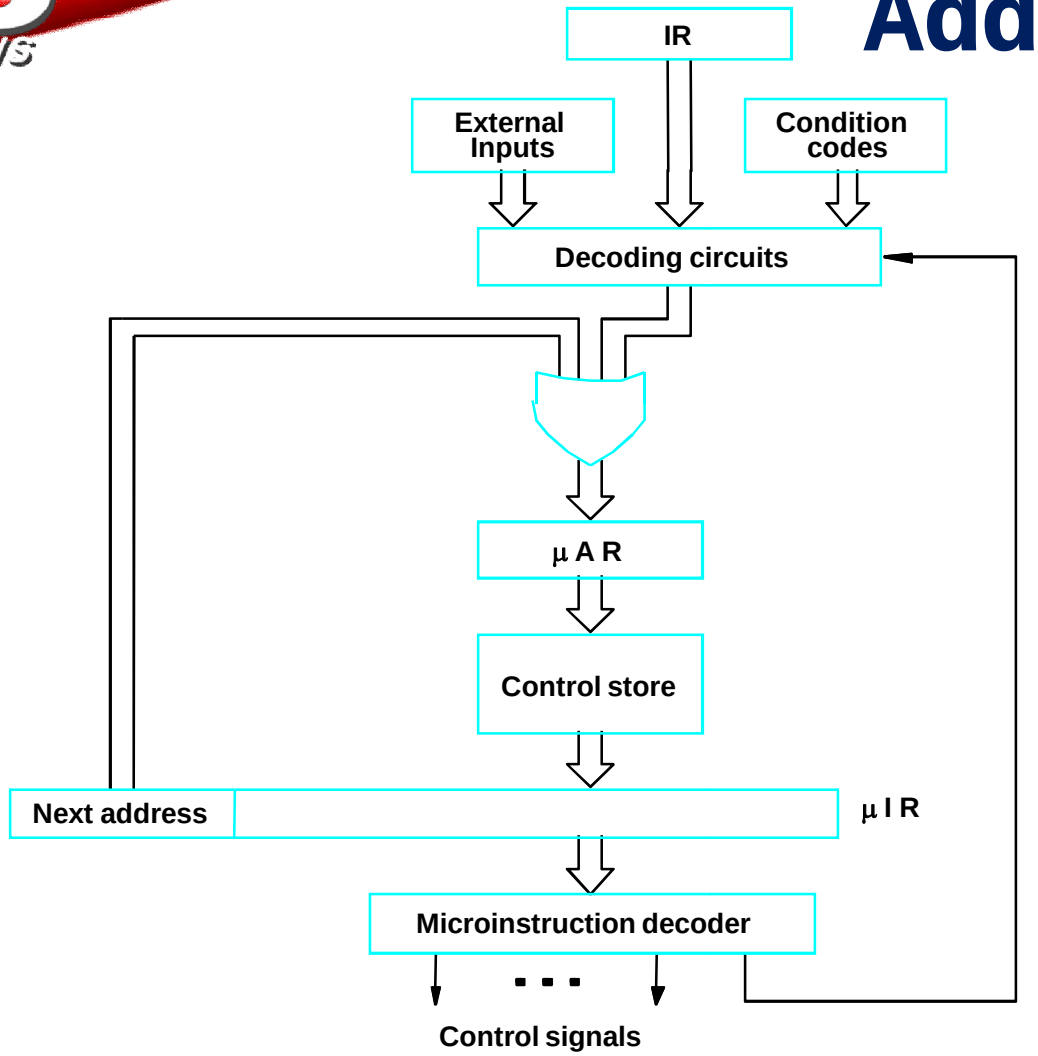


Figure : Microinstruction-sequencing organization.

TEXT BOOK

Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, McGraw-Hill, 6th Edition 2012.

REFERENCES

1. David A. Patterson and John L. Hennessey, “Computer organization and design”, MorganKauffman ,Elsevier, 5th edition, 2014.
2. William Stallings, “Computer Organization and Architecture designing for Performance”, Pearson Education 8th Edition, 2010
3. John P.Hayes, “Computer Architecture and Organization”, McGraw Hill, 3rd Edition, 2002
4. M. Morris R. Mano “Computer System Architecture” 3rd Edition 2007
5. David A. Patterson “Computer Architecture: A Quantitative Approach”, Morgan Kaufmann; 5th edition 2011

THANK YOU